CSE574 Introduction to Machine Learning

Neural Networks 1

Jue Guo

University at Buffalo

February 29, 2024

Outline

Neural Networks

First of all, you already know what a neural network is, and you already know how to build such a model.

- Yes, it's logistic regression! As a matter of fact, the logistic regression model, or rather its generalization for multiclass classification, called the softmax regression model, is a standard unit in a neural network.

A neural network (NN), just like a regression or an SVM model, is a mathematical function:

$$y = f_{NN}(\mathbf{x}).$$

The function f_{NN} has a particular form: it's a nested function.

You have probably already heard of neural network layers. So, for a 3-layer neural network that returns a scalar, f_{NN} looks like this:

$$y = f_{NN}(\mathbf{x}) = f_3\left(\boldsymbol{f}_2\left(\boldsymbol{f}_1(\mathbf{x})\right)\right)$$

In the above equation, f_1 and f_2 are vector functions of the following form:

$$\boldsymbol{f}_l(\mathbf{z}) \overset{\text{def}}{=} \boldsymbol{g}_l\left(\mathbf{W}_l\mathbf{z} + \mathbf{b}_l\right)$$

where l is called the layer index and can span from 1 to any number of layers. The function $\boldsymbol{g}_l$ is called an **activation function**. It is a fixed, usually nonlinear function chosen by the data analyst before the learning is started. The parameters $\mathbf{W}_l$ (a matrix) and $\mathbf{b}_l$ (a vector) for each layer are learned using the familiar gradient descent by optimizing, depending on the task, a particular cost function (such as MSE).

Compared with the equation for logistic regression, where you replace $\boldsymbol{g}_l$ by the sigmoid function, and you will not see any difference. The function f_3 is a scalar function for the regression task, but can also be a vector function depending on your problem.

You may probably wonder why a matrix $\mathbf{W}_l$ is used and not a vector $\mathbf{w}_l$.

- The reason is that $\boldsymbol{g}_l$ is a vector function. Each row $\mathbf{w}_{l,u}$ (u for unit) of the matrix $\mathbf{W}_l$ is a vector of the same dimensionality as $\mathbf{z}$. Let $a_{l,u} = \mathbf{w}_{l,u}\mathbf{z} + b_{l,u}$.

The output of $\boldsymbol{f}_l(\mathbf{z})$ is a vector $\left[g_l\left(a_{l,1}\right), g_l\left(a_{l,2}\right), \ldots, g_l\left(a_{l,size_l}\right)\right]$, where g_l is some scalar function [1], and size $_l$ is the number of units in layer l. To make it more concrete, let's consider one architecture of neural networks called **multilayer perceptron** and often referred to as a **vanilla neural network**.

[1] A scalar function outputs a scalar, that is a simple number and not a vector.

Multilayer Perceptron Example

We have a closer look at one particular configuration of neural networks called **feed-forward neural networks** (FFNN), and more specifically the architecture called a **multilayer perceptron** (MLP).

- As an illustration, let’s consider an MLP with three layers. Our network takes a two-dimensional feature vector as input and outputs a number. This FFNN can be a regression or a classification model, depending on the activation function used in the third, output layer.

The neural network is represented graphically as a connected combination of **units** logically organized into one or more **layers**.

- Each unit is represented by either a circle or a rectangle. The inbound arrow represents an input of a unit and indicates where this input came from. The outbound arrow indicates the output of a unit.

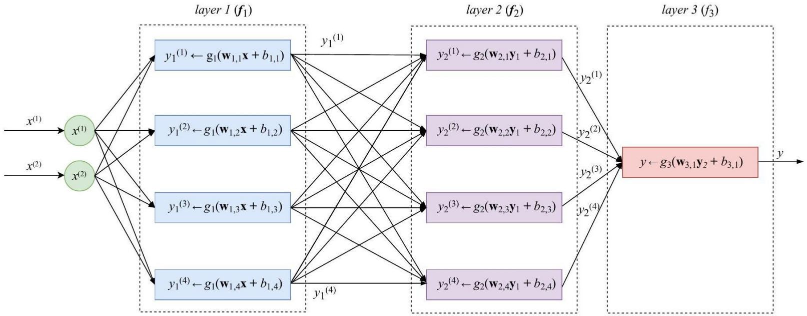

layer 1 (f_1)
layer 2 (f_2)
layer 3 (f_3)
$x^{(1)}$
$x^{(2)}$
$y_1^{(1)} \leftarrow g_1(\mathbf{w}_{1,1}\mathbf{x} + b_{1,1})$
$y_1^{(2)} \leftarrow g_1(\mathbf{w}_{1,2}\mathbf{x} + b_{1,2})$
$y_1^{(3)} \leftarrow g_1(\mathbf{w}_{1,3}\mathbf{x} + b_{1,3})$
$y_1^{(4)} \leftarrow g_1(\mathbf{w}_{1,4}\mathbf{x} + b_{1,4})$
$y_1^{(1)}$
$y_1^{(4)}$
$y_2^{(1)} \leftarrow g_2(\mathbf{w}_{2,1}\mathbf{y}_1 + b_{2,1})$
$y_2^{(2)} \leftarrow g_2(\mathbf{w}_{2,2}\mathbf{y}_1 + b_{2,2})$
$y_2^{(3)} \leftarrow g_2(\mathbf{w}_{2,3}\mathbf{y}_1 + b_{2,3})$
$y_2^{(4)} \leftarrow g_2(\mathbf{w}_{2,4}\mathbf{y}_1 + b_{2,4})$
$y_2^{(1)}$
$y_2^{(2)}$
$y_2^{(3)}$
$y_2^{(4)}$
$y \leftarrow g_3(\mathbf{w}_{3,1}\mathbf{y}_2 + b_{3,1})$
y

Figure: A multi-layer perceptron with two-dimensional input, two layers with four units and one output layer with one unit.

The output of each unit is the result of the mathematical operation written inside the rectangle. Circle units don’t do anything with the input; they just send their input directly to the output.

The following happens in each rectangle unit.

1. Firstly, all inputs of the unit are joined together to form an input vector.
2. Then the unit applies a linear transformation to the input vector, exactly like linear regression model does with its input feature vector.
3. Finally, the unit applies an activation function g to the result of the linear transformation and obtains the output value, a real number.

In a vanilla FFNN, the output value of a unit of some layer becomes an input value of each of the units of the subsequent layer.

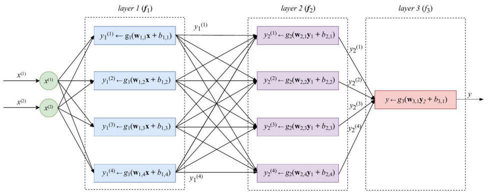
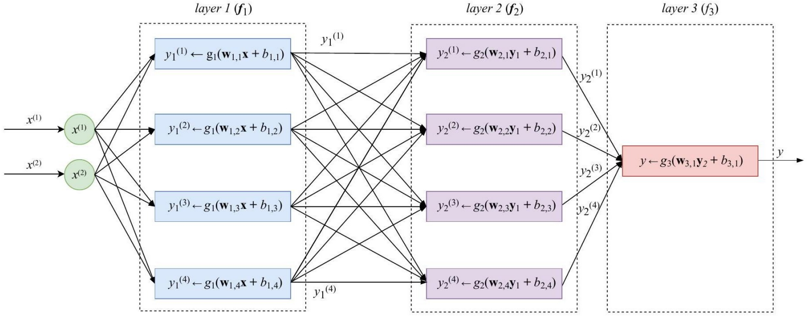

layer 1 (f_1)
layer 2 (f_2)
layer 3 (f_3)
$x^{(1)}$
$x^{(2)}$
$y_1^{(1)} \leftarrow g_1(\mathbf{w}_{1,1}\mathbf{x} + b_{1,1})$
$y_1^{(2)} \leftarrow g_1(\mathbf{w}_{1,2}\mathbf{x} + b_{1,2})$
$y_1^{(3)} \leftarrow g_1(\mathbf{w}_{1,3}\mathbf{x} + b_{1,3})$
$y_1^{(4)} \leftarrow g_1(\mathbf{w}_{1,4}\mathbf{x} + b_{1,4})$
$y_1^{(1)}$
$y_1^{(4)}$
$y_2^{(1)} \leftarrow g_2(\mathbf{w}_{2,1}\mathbf{y}_1 + b_{2,1})$
$y_2^{(2)} \leftarrow g_2(\mathbf{w}_{2,2}\mathbf{y}_1 + b_{2,2})$
$y_2^{(3)} \leftarrow g_2(\mathbf{w}_{2,3}\mathbf{y}_1 + b_{2,3})$
$y_2^{(4)} \leftarrow g_2(\mathbf{w}_{2,4}\mathbf{y}_1 + b_{2,4})$
$y_2^{(1)}$
$y_2^{(2)}$
$y_2^{(3)}$
$y_2^{(4)}$
$y \leftarrow g_3(\mathbf{w}_{3,1}\mathbf{y}_2 + b_{3,1})$
y

Figure: A multi-layer perceptron with two-dimensional input, two layers with four units and one output layer with one unit.

The activation function g_l has one index: l, the index of the layer the unit belongs to. Usually, all units of a layer use the same activation function, but it's not a rule.

- Each layer can have a different number of units.
- Each unit has its parameters $\mathbf{w}_{l,u}$ and $b_{l,u}$, where u is the index of the unit, and l is the index of the layer.

The vector $\mathbf{y}_{l-1}$ in each unit is defined as $\left[y_{l-1}^{(1)}, y_{l-1}^{(2)}, y_{l-1}^{(3)}, y_{l-1}^{(4)}\right]$. The vector $\mathbf{x}$ in the first layer is defined as $\left[x^{(1)}, \ldots, x^{(D)}\right]$.



layer 1 (f_1)
layer 2 (f_2)
layer 3 (f_3)
$x^{(1)}$
$x^{(2)}$
$y_1^{(1)} \leftarrow g_1(\mathbf{w}_{1,1}\mathbf{x} + b_{1,1})$
$y_1^{(2)} \leftarrow g_1(\mathbf{w}_{1,2}\mathbf{x} + b_{1,2})$
$y_1^{(3)} \leftarrow g_1(\mathbf{w}_{1,3}\mathbf{x} + b_{1,3})$
$y_1^{(4)} \leftarrow g_1(\mathbf{w}_{1,4}\mathbf{x} + b_{1,4})$
$y_1^{(1)}$
$y_1^{(4)}$
$y_2^{(1)} \leftarrow g_2(\mathbf{w}_{2,1}\mathbf{y}_1 + b_{2,1})$
$y_2^{(2)} \leftarrow g_2(\mathbf{w}_{2,2}\mathbf{y}_1 + b_{2,2})$
$y_2^{(3)} \leftarrow g_2(\mathbf{w}_{2,3}\mathbf{y}_1 + b_{2,3})$
$y_2^{(4)} \leftarrow g_2(\mathbf{w}_{2,4}\mathbf{y}_1 + b_{2,4})$
$y_2^{(1)}$
$y_2^{(2)}$
$y_2^{(3)}$
$y_2^{(4)}$
$y \leftarrow g_3(\mathbf{w}_{3,1}\mathbf{y}_2 + b_{3,1})$
y

Figure: A multi-layer perceptron with two-dimensional input, two layers with four units and one output layer with one unit.

In multilayer perceptron all outputs of one layer are connected to each input of the succeeding layer. This architecture is called **fully-connected**. A neural network can contain **fully-connected layers**. Those are the layers whose units receive as inputs the outputs of each of the units of the previous layer.

Feed-Forward Neural Network Architecture

If we want to solve a regression or a classification problem, the last (the rightmost) layer of a neural network usually contains only one unit.

- If the activation function g_{last} of the last unit is linear, then the neural network is a regression model.
- If the g_{last} is a logistic function, the neural network is a binary classification model.

The data analyst can choose any mathematical function as $g_{l,u}$, assuming it's differentiable [2]. The latter property is essential for gradient descent used to find the values of the parameters $\mathbf{w}_{l,u}$ and $b_{l,u}$ for all l and u.

- The primary purpose of having nonlinear components in the function f_{NN} is to allow the neural network to approximate nonlinear functions.

Without nonlinearities, f_{NN} would be linear, no matter how many layers it has. The reason is that $\mathbf{W}_l\mathbf{z} + \mathbf{b}_l$ is a linear function and a linear function of a linear function is also linear.

[2] The function has to be differentiable across its whole domain or in the majority of the points of its domain. For example, ReLU is not differentiable at 0 .

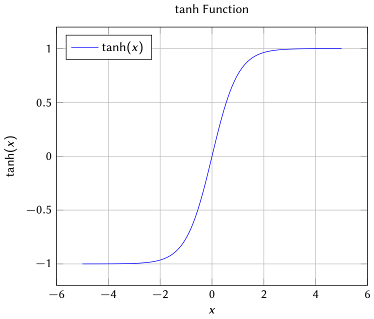
Activation Functions

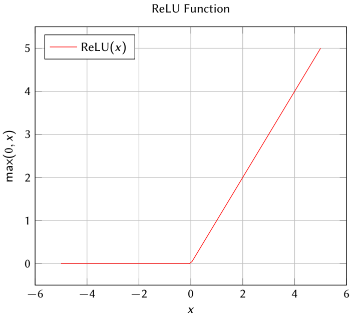
Popular choices of activation functions are the logistic function, already known to you, as well as TanH and ReLU.

- The former is the hyperbolic tangent function, similar to the logistic function but ranging from -1 to 1 (without reaching them).
- The latter is the rectified linear unit function, which equals to zero when its input z is negative and to z otherwise:

$$\mathsf{tanh}(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}},$$

$$\mathsf{relu}(z) = \begin{cases} 0 & \text{if } z < 0 \\ z & \text{otherwise} \end{cases}$$


tanh Function
tanh(x)
1
0.5
0
−0.5
−1
−6
−4
−2
0
2
4
6
x
tanh(x)


ReLU Function
ReLU(x)
5
4
3
2
1
0
−6
−4
−2
0
2
4
6
x
max(0, x)